

Questions in along the project

- 1) What is a Device Tree? Why was it introduced and why is it so important?
 - a. A Device Tree is the hierarchical representation of the devices available in the Operative System for Kernel access mainly.
 - b. This was introduced because We need to understand a standard language to refer to any hardware available for the OS and manipulate it when needed.
- 2) What is a Device Tree Overlay? What can be controlled with it?
 - a. This is a modification in the Device Tree that changes its current state.
 - b. It controls any change of the state of a device. For example, if a device is deactivated, then that devices should disappear from the tree. In our case, we used it to disable Bluetooth in order to allow serial connection to be created on its place.
- 3) Please explain in your report what are the processor fuses and bootstrap resistors and what exactly of the boot process can be controlled with them. Mention a processor (architecture and manufacturer) for which you can control the boot process using both methods.
- 4) Investigate the default location in RAM of the device tree at boot time for the RPI4 and look for information about reserved RAM addresses that U-Boot uses to operate. Describe your findings in the written report and select your **kernel_addr_r** accordingly.
- 5) Investigate a bit more regarding NFS:
 - a. Is the whole file system loaded into RAM when you mount a NFS? If not, which is the criteria followed by the Linux Kernel to load files from the file system into RAM?
 - b. What if we use a Initrd? Would it be completely copied into RAM at boot time?
 - c. What are the benefits and use cases of both of them?

Results

While establishing UART was pretty straightforward, when it came to load the U-Boot and get it running was a bit more difficult. For some reason the KERNEL variable for the crosscompiler script was set to kernel7 instead of kernel8 as suggested in the project guide. First time I skipped the 3.5.1 step since had it installed on the Ubuntu virtual version attached with the project files. Once I got some problems in the following steps I went back to this point and installed it again.

After this little issue, the rest of the steps went pretty well. I was great to see the next text coming out from the UART console on the Ubuntu host:

```

DRAM: 3.9 GiB
RPI 4 Model B (0xc03111)
MMC: mmcnr@7e300000: 1, emmc2@7e340000: 0
Loading Environment from FAT... OK
In: serial
Out: serial
Err: serial
Net: eth0: genet@7d580000
Hit any key to stop autoboot: 0
genet@7d580000 Waiting for PHY auto negotiation to complete.... done
Using genet@7d580000 device
TFTP from server 192.168.8.2; our IP address is 192.168.8.3
Filename 'Image'.
Load address: 0x40000000
Loading: *

```

From time to time, when you disconnect the Ethernet and connect it again, we always got the following error and couldn't get to boot:

```

Loading: *
ARP Retry count exceeded; starting again
U-Boot>

```

However, this is solved by just powering cycle the RPi4, and here is the happy ending:

```

Poky (Yocto Project Reference Distro) 3.0.3 raspberrypi4-64 ttyAMA0
raspberrypi4-64 login: root
root@raspberrypi4-64:~#

```

Which means that we are controlling the RPi4 by means of UART connection ttyAMA0 (on the RPi4) from our Ubuntu host over the ttyUART0 port.

En cuanto a la generación de la receta de Yocto, esta se creó satisfactoriamente, al punto que podemos ver tanto el árbol correctamente generado

```

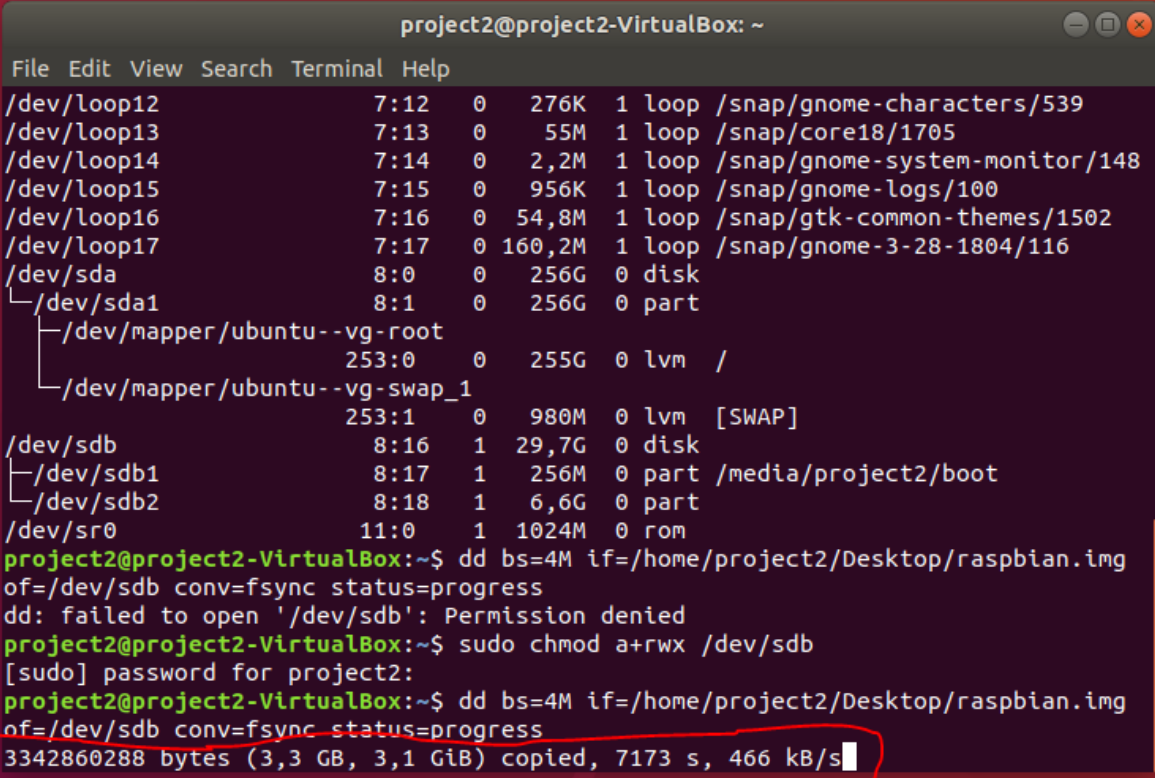
project2@project2-VirtualBox:~$ cd Yocto/poky/meta-tec/
project2@project2-VirtualBox:~/Yocto/poky/meta-tec$ tree
.
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
├── recipes-trapezoidal
│   └── trapezoidal
│       ├── trapezoidal-1.0
│       │   └── trapezoidal-1.0.tar.gz
│       └── trapezoidal-1.0.bb
4 directories, 5 files
project2@project2-VirtualBox:~/Yocto/poky/meta-tec$

```

Así como

Problem Found

In step 6 on page 13, when I run the command indicated to make the microSD flash it always happens (I have restarted the process 3 times) that it stays stuck right where the image I attach in this message shows.



```
project2@project2-VirtualBox: ~
File Edit View Search Terminal Help
/dev/loop12      7:12    0    276K    1 loop  /snap/gnome-characters/539
/dev/loop13      7:13    0     55M    1 loop  /snap/core18/1705
/dev/loop14      7:14    0    2,2M    1 loop  /snap/gnome-system-monitor/148
/dev/loop15      7:15    0    956K    1 loop  /snap/gnome-logs/100
/dev/loop16      7:16    0   54,8M    1 loop  /snap/gtk-common-themes/1502
/dev/loop17      7:17    0  160,2M    1 loop  /snap/gnome-3-28-1804/116
/dev/sda         8:0     0   256G    0 disk
├─/dev/sda1      8:1     0   256G    0 part
│   └─/dev/mapper/ubuntu--vg-root
│       253:0     0   255G    0 lvm    /
│   └─/dev/mapper/ubuntu--vg-swap_1
│       253:1     0   980M    0 lvm    [SWAP]
└─/dev/sdb       8:16    1  29,7G    0 disk
    ├─/dev/sdb1   8:17    1   256M    0 part  /media/project2/boot
    └─/dev/sdb2   8:18    1    6,6G    0 part
/dev/sr0        11:0    1  1024M    0 rom
project2@project2-VirtualBox:~$ dd bs=4M if=/home/project2/Desktop/raspbian.img
of=/dev/sdb conv=fsync status=progress
dd: failed to open '/dev/sdb': Permission denied
project2@project2-VirtualBox:~$ sudo chmod a+rwx /dev/sdb
[sudo] password for project2:
project2@project2-VirtualBox:~$ dd bs=4M if=/home/project2/Desktop/raspbian.img
of=/dev/sdb conv=fsync status=progress
3342860288 bytes (3,3 GB, 3,1 GiB) copied, 7173 s, 466 kB/s
```

I assigned 8GB of RAM and 4 cores (the maximum I can) to Ubuntu 18.04 running on Virtual Box 6.0.22. All the previous steps have been followed without problems.

How I tackled this issue

I installed Etcher on Windows and flashed the Raspbian image on the microSD from Windows. Afterwards realized that USB 3.0 was not enabled in the Virtual Box.